MODELOS DE COMPUTACIÓN PARALELA

Printista A.Marcela, Piccoli M.Fabiana
Universidad Nacional de San Luis
Ejército de los Andes 950
(5700) San Luis
{mprinti, mpiccoli}@unsl.edu.ar
Argentina

Resumen

Desde los primeros días de la computación ha sido claro que, tarde o temprano, surgiría una demanda de mayor poder de computación que el que brinda la computación secuencial. A pesar de que la *Computación Paralela* ha surgido como el candidato natural para cubrir la insistente demanda de mayor performance y a pesar de la mayor disponibilidad de máquinas paralelas, aún no ha logrado imponerse como paradigma corriente de computación.

¿Cuál podría ser la clave que impulsará el desarrollo de computaciones paralelas en el futuro?

Hay tres candidatos obvios:

- el hardware
- el software
- un modelo de computación intermediario

Mucho se ha discutido de las dos primeras opciones [Qui94][Jaj92][Wil96]. Actualmente se encuentra disponible una importante cantidad de estudios teóricos y prácticos sobre computación paralela realizados sobre distintas arquitecturas paralelas con distintos lenguajes y herramientas de programación. Sin embargo, pocos trabajos son los que han cubierto los dos objetivos principales de la computación paralela; por un lado desarrollar software independiente de la arquitectura y tecnologías existentes, y por otro, desarrollar algoritmos que deban obtener la máxima performance de la arquitectura subyacente. Lo cierto es que estos objetivos, en primera instancia se presentan como contradictorios.

Uno de los objetivos de nuestra línea de investigación en Sistemas Paralelos es el estudio de la validez de los modelos para predecir la performance de programas paralelos en las arquitecturas actuales. Este objetivo ha surgido basándose en nuestra fuerte argumentación que tener un modelo de computación paralela intermediario es la principal clave de progreso a partir de nuestra posición presente.

La comunidad de computación serial dispone de un Modelo de Computación ampliamente aceptado, el modelo de *Von Neumann*, expresado elegantemente en la *Máquina de Acceso Random (RAM)* [For78]. Este modelo ha actuado exitosamente como modelo de computación subyacente proporcionado consistencia y coordinación entre los diseñadores de algoritmos, constructores de arquitecturas y expertos en lenguajes.

A pesar de las justificaciones que existen en tener un modelo de computación, en la computación paralela, ningún modelo unificado ha obtenido el mismo éxito.

Para que la computación paralela llegue a ser una forma normal de computación se requiere un modelo que juegue el mismo rol que cumple el de *Von Neumann* en la computación secuencial: ofrecer un camino bien definido entre el software y el hardware[Juu96][Mag95].

El gran reto está en desarrollar un modelo unificado que consiga englobar las diferentes propuestas existentes. Muchos científicos son escépticos y opinan que la modelización en la computación paralela

aún es controversial y caótica. No obstante la comunidad científica ha generado un amplio rango de modelos con el objetivo de dilucidar los requerimientos de un paradigma unificado.

Con el objetivo de abrir un marco de discusión, para este trabajo, se han seleccionado y recopilado un amplio rango de modelos de computación paralela, a fin de conocer las características y objetivos esenciales de cada uno, de analizar el rol que estos modelos cumplen en los algoritmos, en los lenguajes y en los diseños de máquinas actuales y futuras.

En primer lugar se consideró el modelo *Parallel Random Access Machine, PRAM* [Har94]. Este surge basado en el éxito del modelo *RAM*. Aunque existen variaciones entre las definiciones de *PRAM*, el modelo estándar es una computadora *MIMD*, donde cada procesador puede ejecutar su propio flujo de instrucciones. Cada procesador puede acceder a cualquier ubicación en un tiempo de acceso independiente de cual sea la posición a acceder. Dejando de lado la imprecisión sobre computadoras paralelas prácticas, el modelo ofrece un marco útil para el diseño y análisis. El objetivo del modelo es que el diseñador explote al máximo el paralelismo inherente en una tarea dada. En este sentido, *PRAM* provee una medida de la complejidad de tiempo ideal [Agg90]. Las críticas asociadas a este modelo, argumentan que las características no representadas relacionadas al costo de usar un recurso, conlleva al abuso del mismo. En otras palabras, los aspectos que el modelo oculta sirven para distorsionar el proceso de diseño llevando a una solución práctica ineficiente.

La noción de modelo de puente (*bridging model*), fue efectivamente capturado por Valiant [Val90]. En sus trabajos, el autor presenta una máquina abstracta la cual provee un paradigma de diseño consistente y unificado para facilitar el diseño de algoritmos paralelos portables y su traslación a un programa. El modelo, *Bulk Synchronous Parallel, BSP*, posee una memoria distribuida y asume dos parámetros. El primero, *l*, refleja el costo de invocar una operación de sincronización. Esto implica una latencia de comunicación ya que los accesos a memoria remota no son efectivos hasta después de una operación de sincronización. El segundo parámetro, *g*, representa las limitaciones de *bandwidth*. Se requiere que los mensajes sean enviados a lo sumo una vez cada *g* operaciones aritméticas [Juu96a].

El modelo *BSP* sobresale por aquellas características que no incorpora, por ejemplo no incorpora *overhead* cuando un mensaje es ingresado a la red [Ski96]. Tampoco incluye ningún dato acerca de la topología lo cual remueve la necesidad de incluir patrones de comunicación diferenciados y realizar optimizaciones basadas en la topología [Gou96].

Otro ejemplo de un modelo de puente es *LogP* [Cul96][Ian98], el cual realiza más énfasis en los atributos de las máquinas. Este modelo aunque esta muy relacionado al modelo *BSP*, es distinto por dos razones. Primero, el modelo considera comunicación asincrónica. El parámetro *l* es usado estrictamente como una medida de la latencia de mensajes. Segundo, el modelo agrega un parámetro más, *o*, el cual representa el tiempo necesario para introducir o recuperar un mensaje a/desde la red. Este parámetro mide un tiempo muerto, ya que son ciclos de procesador ocioso que no pueden ser ocultados con ninguna medida de latencia.

Los modelos estándares *PRAM* y *BSP* poseen un patrón de ejecución rígido en el cual todos los procesadores son sincronizados por un *clock* global. Una variante, el modelo *Oblivious BSP* [Gon00], suaviza esta restricción. El incentivo subyacente de este modelo es sincronizar sólo cuando es necesario. El modelo *BSP* es así extendido con un mecanismo de sincronización de costo cero, el cual es usado cuando el número de mensajes a recibir es conocido. Al realizar una predicción de performance de programas con sincronización oblivia, siguiendo un estilo *BSP*, se produce una pérdida de exactitud en la predicción. Como una consecuencia de esto el modelo *Oblivious BSP* propone un nuevo modelo de complejidad para tratar con este nuevo tipo de barreras (*oblivious barrier*).

Conclusiones

Los modelos de computación presentados en esta trabajo fueron elegidos como representantes de un numeroso conjunto de modelos abstractos de computación paralela. Este subconjunto es suficiente testimonio de que no existe aún consenso y se consolida más la necesidad de un paradigma unificado.

Dentro de este conjunto de modelos podría encontrarse el modelo consistente buscado. El conjunto de modelos incluye como parámetros: Paralelismo computacional *(P)*, Latencia de comunicación *(L)*, *Overhead* de comunicación, *Bandwdith*, Sincronización, Jerarquías de memorias y Topologías de red. Estas características principalmente reflejan la perspectiva de investigadores cuyo objetivo primario es el diseño eficiente de algoritmos.

Evaluados separadamente ningún modelo parece ser aceptable, pero evaluados en un conjunto, todos parecen hacer énfasis y coincidir en un pequeño número de características. Consideramos que, para una disciplina que evolucionada día a día es un apreciable progreso.

Bibliografía

[Agg90] Aggarwal A, Chandra A.K., Snir M. *Communication complexity of PRAMs*. Teorical Computer Science, 71:3-28, 1990.

[Cul96] Culler D.E., Karp R., Patterson D., Sahay A., Santos E., Schauser K.E., Subramonian R., Eicken T. *LogP, a Practical Model of Parallel Computation*. Communications of the ACM, Vol. 39, No.11. Nov. 1996.

[For78] Fortune S., Wyllie, J., *Parallelism in Random Access Machines*. Proc. 2nd ACM Annual Symp. on

[Gon00] Gonzalez J.A. , Leon C., Piccoli F., Printista M., Roda J.L., Rodriguez C., Sande F. *Oblivious BSP*.

[Gou96] Goudreau, M., Lang, K., Rao, S. *Towards Efficiency and Portability: Programming with the BSP model*. Proc. SPAA'96: 8th Annual ACM SPAA, 1-12, 1996

[Har94] Harris, T.J., *A Survey of PRAM Simulation Techniques*. ACM Computing Surveys, Vol. 26, No.2. pp. 187-206. June 1994.

[Ian98] Iannello G., Lauria M., Mercolino. *LogP Performance Characterization of Fast Messages atop Myrinet.* Porc. of the 6 th Euromicro Workshop on Parallel and Distributed Processing. Madrid. Jan. 1998.

[Jaj92] JáJá J. *An Introduction to Parallel Algorithms*, Addison-Wesley, 1992.

[Juu96] Juurlink, B.H.H., Wijshoff, H.A.G. *A Quantitative Comparison of Parallel Computation Models.* Proc. SPAA'96. 1996.

[Juu96a] Juurlink, B.H.H., Wijshoff, H.A.G. *Communication Primitives for BSP Computers.* Information Proceeding Letters 58. Pp. 303-310. 1996.

[Mag95] Maggs B.M. , Matheson L., Tarjan R. *Models of Parallel Computations: A survey and Synthesis.* Proceedinhg of the 28 th. Hawaii international conference on Systems Science. IEEEPress. 1995.

[Qui94] Quinn M.- *Parallel Computing. Theory and Practice*. Second Edition. McGraw-Hill, Inc. 1994.

[Ski96] Skillcorn, D.B., Hill, J., McColl, W.F. *Questions and Answers about BSP*. Oxford University Computing Laboratory. Report PRG-TR-15-96. 1996.

[Val90] Valiant L.G.. *A Bridging Model for Parallel Computation*. Communications of the ACM, 33(8): 103-111, 1990.

[Wil96] Wilkinson B. & Allen M., *Parallel programming: Techniques and Application using Networked Workstations*, Prentice-Hall. 1996.